LEVEL

3

Communications Research Centre

6

AN ITERATIVE TECHNIQUE FOR DESIGNING FINITE IMPULSE RESPONSE CHEBYSHEV MTI FILTERS WITH NONLINEAR DELAY

by

10 R.W. Herring

14 CRC-TN-691

9

DEPARTMENT OF COMMUNICATIONS
MINISTÈRE DES COMMUNICATIONS

CRC TECHNICAL NOTE NO. 691

This work was sponsored by the Department of National Defence, Research and Development Branch under Project No. 33C69.

12 58p.

11

CANADA

OTTAWA, MARCH 1978

(3)

COMMUNICATIONS RESEARCH CENTRE

DEPARTMENT OF COMMUNICATIONS
CANADA

AN ITERATIVE TECHNIQUE FOR DESIGNING FINITE IMPULSE RESPONSE CHEBYSHEV MTI FILTERS WITH NONLINEAR PHASE DELAY

by

R.W. Herring

(Radio and Radar Research Branch)

CRC TECHNICAL NOTE NO. 691

March 1978
OTTAWA

This work was sponsored by the Department of National Defence, Research and Development Branch under Project No. 33C69.

TABLE OF CONTENTS

AN ITERATIVE TECHNIQUE FOR DESIGNING FINITE IMPULSE RESPONSE CHEBYSHEV MTI FILTERS WITH NONLINEAR PHASE DELAY

by

R.W. Herring

ABSTRACT

A technique for designing finite-impulse-response (FIR) equiripple high-pass digital filters with nonlinear phase, suitable for use in radar moving-target-indication (MTI) systems, is described. The optimization of such filters in both the time and frequency domains is discussed, and comparisons are made between the spectral performance of nonlinear-phase filters and that of linear-phase filters designed to meet the same stopband bandwidth and attenuation specifications. In 92 out of the 94 cases examined, the transition bandwidth between the stopband is narrower for the nonlinear-phase filter. In the MTI application, this characteristic makes it possible to detect targets of lower radial velocity. Listings of Fortran programs for designing these nonlinear-phase filters are included.

1. INTRODUCTION

The object of this note is to describe a technique for designing finite-impulse-response (FIR) digital filters with nonlinear phase-delays for use in radar moving-target-indication (MTI) systems. The optimization of such filters in both the time and frequency domains is discussed, and comparisons are made between the spectral performance of nonlinear-phase filters and that of linear-phase filters designed to meet the same stopband bandwidth and attenuation specifications. In 92 out of 94 cases examined, the transition bandwidth between the filter stopband and passband is narrower for the nonlinear-phase filter. In the MTI application, this characteristic makes it possible to detect targets of lower radial velocity.

It is well known from digital signal theory (e.g., [1]) that delaying a sampled data sequence by a fixed amount imposes a shift or delay in the

relative phases of its spectral components which is linear with frequency. One of the great advantages of FIR filters is that they can be designed to have such a linear phase-delay, so that their only effects are to modify the magnitudes of the spectral components and to delay the sequence by a fixed amount.

In contrast, infinite-impulse-response (IIR) filters and nonlinear-phase FIR filters have nonlinear phase versus frequency characteristics. The use of nonlinear-phase filters may increase the complexity of any coherent post MTI signal processing, but since the phase characteristics of these filters are deterministic, any undesirable effects due to the phase nonlinearities can be compensated.

Houts and Burlage [2] have described the advantages to be derived by using Chebyshev equiripple FIR filters in MTI systems, and they have published computer programs [3] for the design and evaluation of Chebyshev filters having linear phase-delay. Their design procedure is based on a computer program for designing equiripple FIR linear phase digital filters [4] using the Remez exchange algorithm.

2. BENEFITS OF NONLINEAR PHASE

The removal of the linear-phase constraint can result in improved MTI performance in both the spectral and the time domains. The parameters of interest in MTI filter design and defined in Table 1 and depicted in Figure 1. The standard definitions of equiripple FIR filter parameters (e.g., [5]) are defined in Table 2 and depicted in Figure 2. Note that the standard definitions refer to a filter designed to have unity gain in the passband, whereas for MTI applications, it may be desirable to have non-unity passband gain in order to have 0 dB white noise power gain and/or 0 dB minimum passband gain.

The improved spectral performance obtainable from nonlinear-phase equiripple FIR filters relative to linear-phase equiripple FIR filters can be realized in three different ways. First, smaller ripples in either the stopband or passband ripples (or both) can be achieved for given values of f_{STOP}, f_{PASS} and N. Smaller passband ripples mean increased A_{SB} and thus greater clutter suppression. Decreased R_{PB} means that a higher detection threshold can be used without a loss of target visibility due to signal attenuation in the filter passband.

Second, the transition band, or the band of frequencies between f_{STOP} and f_{PASS}, can be made narrower. Such a narrowing of the transition band is useful when enhanced low-velocity target visibility is desired in the presence of stationary clutter of finite bandwidth.

Third, the performance of a given linear-phase filter can be approximated, except for phase, by a nonlinear-phase filter with smaller N. Thus enhanced incoherent integration gain can be achieved from a given fixed number of radar pulses, since a greater number of independent filtered output pulses are then available for integration [3].

TABLE 1

Definitions of Equiripple MTI Filter Parameters

N	number of weights in filter impulse response
$h(n)$	$(0 \leqslant n \leqslant N-1)$ filter impulse response
f	Doppler frequency (Hz)
$\|H(f)\|$	absolute magnitude of the MTI filter response
$H(f)$	complex magnitude of the MTI filter response
f_{STOP}	frequency of upper edge of filter stopband (Hz)
f_{PASS}	frequency of lower edge of filter passband (Hz)
f_{PR}	radar pulse-repetition frequency (Hz)
R_{PB}	peak-to-peak gain ripple in filter passband (dB)
A_{SB}	minimum filter attenuation – measured from bottoms of passband ripples to peaks of stopband ripples (dB)
G_{WNP}	white-noise-power gain (dB)
G_{PBM}	minimum gain in filter passband (dB)

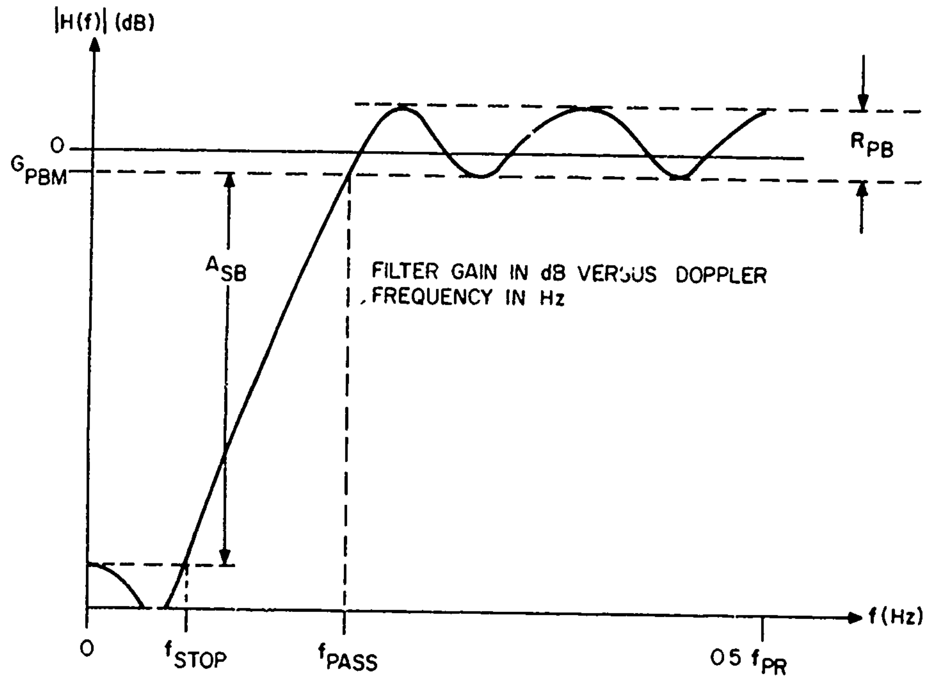

Figure 1. Definition of Equiripple MTI Filter Parameters

TABLE 2

ϵ_1 amplitude of passband ripple (linear)

ϵ_2 amplitude of stopband ripple (linear)

(other definitions as in Table 1)

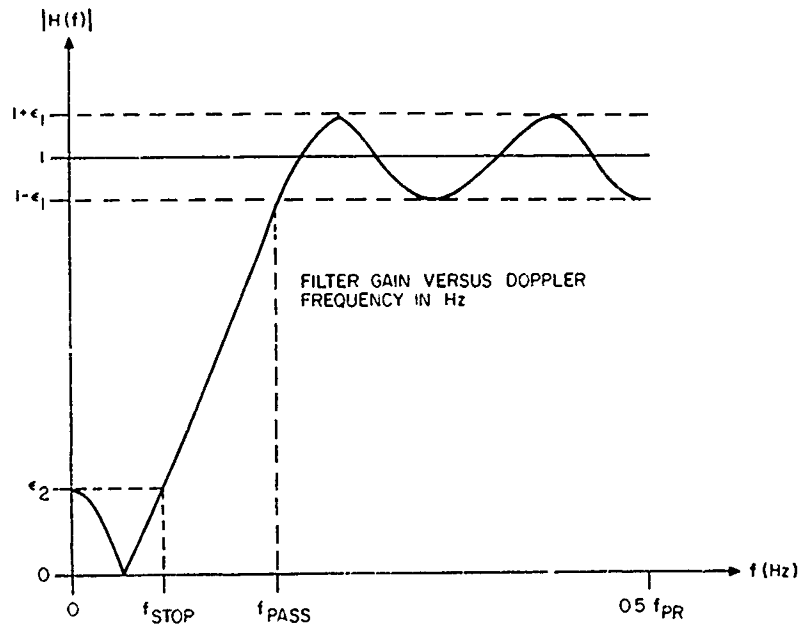

Figure 2. Standard Definition of Equiripple High-Pass Filter Parameters

In Tables 3 and 4 there can be found examples showing each of the three forms of benefit described above. The filters summarized in Table 3 were designed to have A_{SB} = 50 dB and values of f_{STOP} ranging from 50 Hz to 400 Hz. Those summarized in Table 4 were designed to have A_{SB} = 30 dB and values of f_{STOP} ranging from 100 Hz to 400 Hz. In both tables f_{PR} = 2500 Hz. For some sets of design parameters no results are shown, since such filters could not be designed subject to the constraints G_{WNP} = 0 dB and G_{PBM} = 0 dB (*vide* Tables 3a and 3e).

The relaxation of the linear-phase constraint also allows some latitude in the choice of the filter impulse-response function (IRF), because there are several possible IRFs corresponding to a particular spectral amplitude function. Each of these IRFs corresponds to a different phase-delay characteristic, but since phase delay is of no concern, it now becomes possible to select a particular IRF on the basis of its time domain characteristics. Three possible criteria for this selection are: (I) a mini-max criterion, which attempts to equalize the magnitudes of the IRF weights by selecting that IRF having the smallest value of the ratio of the largest to smallest weights; (II) a criterion which minimizes susceptibility to numerical overflow by selecting that IRF having the minimum absolute sum of its weights; or (III) another criterion which attempts to equalize the magnitude of the IRF weight by selecting that IRF having the minimum variance of the magnitudes of its weights. Criteria I and III should therefore select IRFs which

are less susceptible to disruption by impulsive noise, whereas Criterion II is based on minimizing susceptibility to numerical overflow. The evaluation of the suitability of these criteria or the proposal of others remains as a topic requiring further investigation.

TABLE 3a

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	477	2.71	–	–	–	–
6	537	3.30	390	2.12	147	1.18
7	392	2.48	376	2.07	16	0.41
8	421	2.52	347	1.93	74	0.59
9	386	2.16	319	1.79	67	0.37
10	350	2.05	292	1.64	58	0.41
11	352	2.00	270	1.51	82	0.49
12	301	1.73	251	1.40	50	0.33
13	317	1.82	234	1.30	83	0.52
14	265	1.51	219	1.22	46	0.29
15	286	1.65	206	1.14	80	0.51
16	237	1.34	195	1.08	42	0.26

$f_{STOP} = 25$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 50$ dB $G_{WNP} = 0$ dB $G_{PBM} = 0$ dB

TABLE 3b

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	718	4.30	576	3.48	142	0.82
6	539	3.33	482	2.87	57	0.46
7	556	3.44	416	2.45	140	0.99
8	423	2.54	367	2.14	56	0.40
9	455	2.80	332	1.90	123	0.90
10	352	2.07	300	1.72	52	0.35
11	386	2.35	282	1.60	104	0.75
12	303	1.76	280	1.60	23	0.16
13	337	2.03	274	1.57	63	0.46
14	288	1.66	263	1.52	25	0.14
15	300	1.79	252	1.46	48	0.33
16	284	1.65	240	1.39	44	0.26

$f_{STOP} = 50$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 50$ dB $G_{WPN} = 0$ dB $G_{PBM} = 0$ dB

TABLE 3c

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	793	5.38	591	3.67	202	1.71
6	542	3.37	532	3.37	10	0.00
7	583	3.78	511	3.13	72	0.65
8	537	3.33	462	2.84	75	0.49
9	469	2.96	417	2.56	62	0.40
10	477	3.01	380	2.31	97	0.70
11	397	2.47	349	2.11	48	0 36
12	419	2.64	332	2.01	87	0.53
13	346	2.13	330	2.00	16	0.13
14	372	2.33	321	1.95	51	0.28
15	340	2.09	308	1.88	32	0.21
16	335	2.09	293	1.79	42	0.30

$f_{STOP} = 100$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 50$ dB $G_{WNP} = 0$ dB $G_{PBM} = 0$ dB

TABLE 3d

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{FB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	810	5.65	807	5.61	3	0.04
6	818	5.76	648	4.28	170	1.40
7	654	3.55	634	4.16	20	-0.61
8	627	4.26	580	3.81	47	0.45
9	617	4.14	522	3.40	95	0.74
10	519	3.45	479	3.30	40	0.15
11	536	3.60	479	3.10	57	0.50
12	493	3.25	458	2.97	35	0.28
13	471	3.14	431	2.80	40	0.34
14	472	3.14	407	2.64	65	0 50
15	423	2.80	405	2.62	18	0.18
16	436	2.90	397	2.58	39	0.32

$f_{STOP} = 200$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 50$ dB $G_{WNP} = 0$ dB $G_{PBM} = 0$ dB

TABLE 3e

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	–	–	–	–	–	–
6	–	–	–	–	–	.
7	–	–	871	6.66	–	–
8	873	6.65	789	4.89	84	1.76
9	815	5.58	754	5.50	61	0.08
10	716	5.28	684	5.58	32	-0.30
11	723	5.35	686	4.92	37	0.43
12	703	5.16	646	4.61	57	0.55
13	636	5.39	636	4.52	0	0.87
14	647	4.74	618	4.39	29	0.35
15	640	4.68	596	4.07	44	0.61
16	607	3.97	593	4.20	14	-0.23

$f_{STOP} = 400$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 50$ dB $G_{WNP} = 0$ dB $G_{PBM} = 0$ dB

TABLE 4a

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	492	2.65	489	2.78	3	-0.13
6	534	3.27	455	2.61	79	0.66
7	485	2.79	407	2.35	78	0.44
8	422	2.53	365	2.12	57	0.41
9	428	2.53	330	1.91	98	0.62
10	353	2.09	302	1.75	51	0.34
11	374	2.23	279	1.62	95	0.61
12	306	1.80	270	1.56	36	0.24
13	332	1.98	268	1.55	64	0.43
14	278	1.49	262	1.53	16	-0.04
15	298	1.78	253	1.48	45	0.30
16	275	1.61	243	1.43	32	0.18

$f_{STOP} = 100$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 30$ dB $G_{WNP} = 0$ dB $G_{PBM} = 0$ dB

TABLE 4b

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	766	4.97	594	3.71	172	1.26
6	549	3.46	503	3.64	46	-0.18
7	479	3.73	505	3.12	74	0.61
8	521	3.28	469	2.91	53	0.37
9	474	3.03	428	2.67	46	0.36
10	479	3.05	394	2.47	85	0.58
11	409	2.61	387	2.42	22	0.19
12	428	2.74	380	2.39	48	0.35
13	395	2.52	366	2.31	29	0.21
14	386	2.48	349	2.21	37	0.27
15	384	2.46	334	2.12	50	0.34
16	353	2.28	331	2.11	22	0.17

$f_{STOP} = 200$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 30$ dB $G_{WNP} = 0$ dB $G_{PBM} = 0$ dB

TABLE 4c

Comparative Examples of Equiripple Finite-Impulse-Response MTI Filter Characteristics

	LINEAR PHASE		NONLINEAR PHASE			
Number of Weights N	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Passband Lower Edge (Hz) f_{PASS}	Passband Ripple (dB) R_{PB}	Reduction of Passband Lower Edge (Hz)	Reduction in Passband Ripple (dB)
5	822	5.85	822	5.84	0	0.01
6	829	5.95	759	4.66	70	1.29
7	773	5.06	697	4.85	76	0.21
8	600	4.66	630	4.97	-30	-0.31
9	669	4.75	630	4.35	39	0.40
10	645	4.55	593	4.10	52	0.45
11	588	4.17	579	4.00	9	0.17
12	597	4.24	567	3.92	30	0.32
13	584	4.15	543	3.77	41	0.38
14	548	3.91	542	3.75	6	0.16
15	555	3.96	530	3.68	25	0.28
16	548	3.90	518	3.55	30	0.35

$f_{STOP} = 400$ Hz $f_{PR} = 2500$ Hz $A_{SB} = 30$ dB $G_{WNP} = 0$ dB $G_{PBM} = 0$ dB

3. OUTLINE OF THE DESIGN PROCEDURE

The technique for designing nonlinear-phase FIR filters is based on a procedure first suggested in [6]. This procedure involves the design of a prototype linear-phase FIR filter having an IRF $h_p(n)$ of length (2N-1) and an amplitude response in the frequency domain $H_p(f)$ equal to the square of the magnitude of the desired frequency response. It is shown below that this prototype filter cannot be designed directly from the desired parameters of the nonlinear-phase filter, but that an iterative technique must be used.

A property of linear-phase FIR filters with real-valued IRFs is that if z_o is a zero of the IRF, then so are $z_o{}^{-1}$, z_o^* and $(z_o{}^{-1})^*$ where * denotes complex conjugate ([1]; page 159). Note that if z_o is real, then $z_o = z_o^*$ and also, if $|z_o| = 1$, then $z_o{}^{-1} = z_o^*$, so that roots can be real and single, or occur in conjugate pairs (if $|z_o| = 1$) or in reciprocal pairs (if z_o is real), or in conjugate-reciprocal quartets (if $|z_o| \neq 1$ and z_o complex). In general, of the 2(N-1) zeroes of the (2N-1)-length prototype filter, there are k pairs of reciprocal real zeroes, ℓ quartets of conjugate reciprocal zeroes and $m = (N-k-2\ell-1)$ pairs of double zeroes on the unit circle. To extract an IRF h(n) of length N, it is necessary to discard one zero of each of the m pairs of double zeroes, one zero of each of the k pairs of reciprocal zeroes, and one pair of conjugate zeroes from each of the ℓ quartets of conjugate-reciprocal zeroes. This leaves a set of (N-1) zeroes which is expanded to produce a real-valued IRF of length N.

A set of M IRFs of length N can thus be derived from the (2N-1)-length prototype filter, where

$$M = 2^{(k+\ell-1)}$$

Note that one zero or pair of conjugate zeroes can be arbitrarily chosen to lie inside or outside the unit circle, since the only effect of this choice is to reverse the IRFs in the time domain; i.e., h(n) is replaced by h(N-1-n) for $0 \leq n \leq N-1$.

An upper bound on M as a function of N is given by

$$M_{UB} = 2^{\{[(N-2)/2]-1\}}$$

where [x] denotes the largest integer less than x, so it can be seen that the set of IRFs to be examined can contain of order 2^{10} for N=24. Hence the optimization techniques described above can become quite time-consuming for filters of such length. This limitation, however, should not preclude the use of these procedures for designing filters of the lengths usually considered for MTI applications (e.g., $N \leq 20$; so that $M \leq 256$).

4. DESIGNING THE PROTOTYPE FILTER

In order to make use of the standard filter design algorithms, it is first necessary to define the filter ripple parameters (R_{PB},A_{SB}) of interest to radar MTI designers in terms of the standard linear ripple parameters

$(\varepsilon_1, \varepsilon_2)$. It can easily be shown from Figures 1 and 2 that

$$\varepsilon_1 = \frac{10^{(R_{PB}/20)} - 1}{10^{(R_{PB}/20)} + 1} \tag{1}$$

and

$$\varepsilon_2 = \frac{1 - \varepsilon_1}{10^{(A_{SB}/20)}} \, . \tag{2}$$

In the standard design procedure for linear-phase FIR filters [3], if N, f_{STOP}, f_{PASS} and f_{PR} are specified, then only the ratio

$$W = \varepsilon_1/\varepsilon_2 \tag{3}$$

can be specified as a free parameter. This restriction can be circumvented by allowing f_{PASS} to be varied in an iterative manner until that value for f_{PASS} is found which gives the desired values for ε_2 and hence ε_1 [3].

A similar iterative procedure is necessary in the design of nonlinear-phase FIR filters. For the design of the linear-phase prototype filter $H_o(f)$ (see Figures 3(a)-3(c)), it is necessary to specify 2N-1, f_{STOP}, f_{PR}, δ_1 and δ_2, where δ_1 and δ_2 have to be specified in terms of ε_1 and ε_2. The outline of the scheme for relating the δ's and ε's is pictured in Figures 3(a)-3(c) and is similar to that of [6] except for one detail pointed out below. For clarity the scheme is described progressing from the original prototype filter $H_o(f)$ (Figure 3(a)) to the intermediate filter $H_I(f)$ (Figure 3(b)) to the final prototype filter $H_p(f) = |H(f)|$ (Figure 3(c)). In practice, the actual progression is from specifying $H_p(f)$ to specifying $H_o(f)$ in terms of $H_p(f)$, since $H_o(f)$ is the filter which is actually designed using the algorithm of [4].

$H_I(f)$ is related to $H_o(f)$ by the transformation

$$H_I(f) = H_o(f) + \delta_2 . \tag{4}$$

In the time domain this is equivalent to

$$h_I(n) = \begin{cases} h_o(n), & 1 \le |n| \le N-1 \\ h_o(n) + \delta_2, & n = 0. \end{cases} \tag{5}$$

$H_p(f)$ is in turn related to $H_I(f)$ by the transformation

$$H_p(f) = K\, H_I(f) \tag{6}$$

which becomes in the time domain

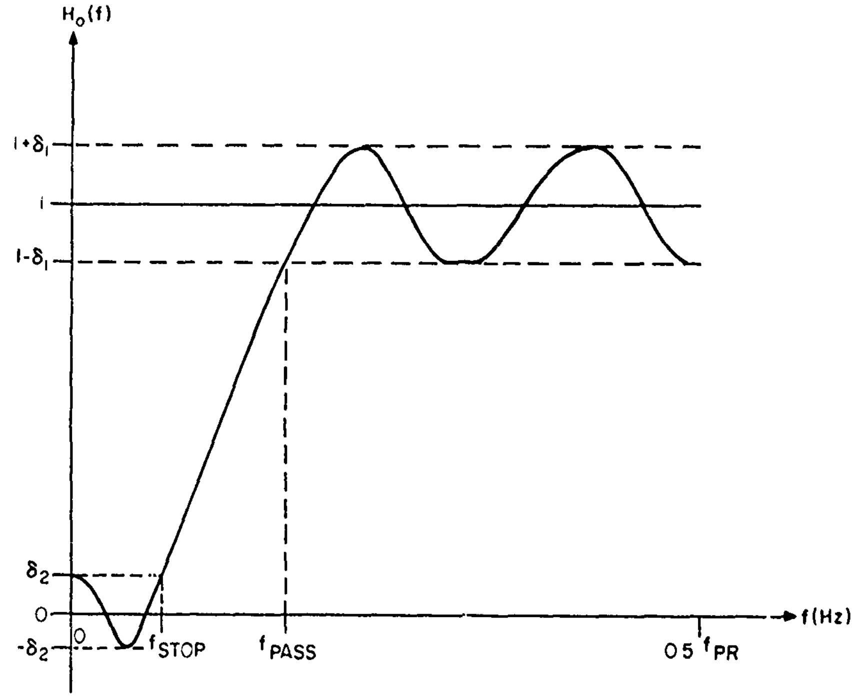

Figure 3(a). Original Linear-Phase Prototype Filter Frequency Response

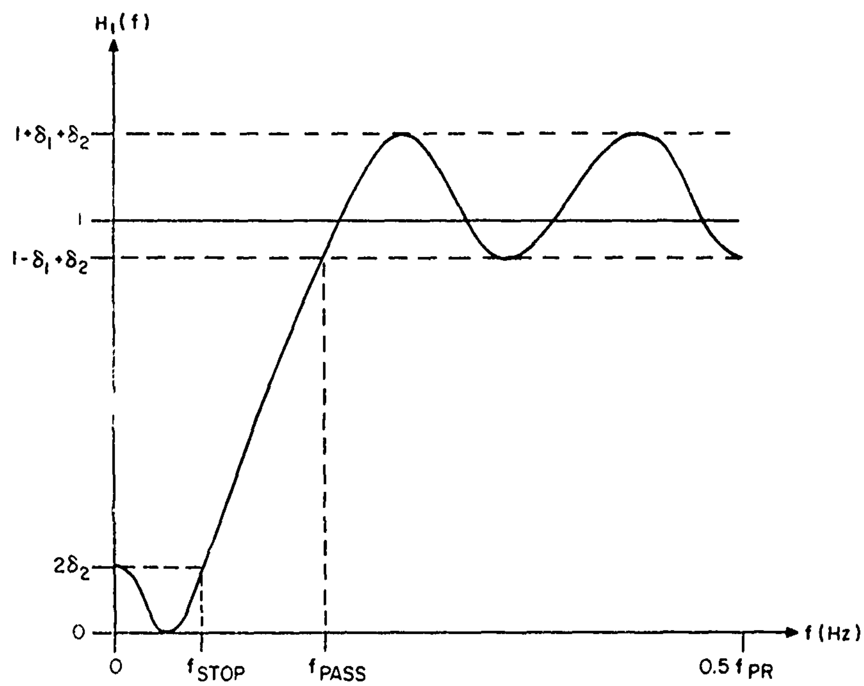

Figure 3(b) Intermediate Linear-Phase Prototype Filter Frequency Response

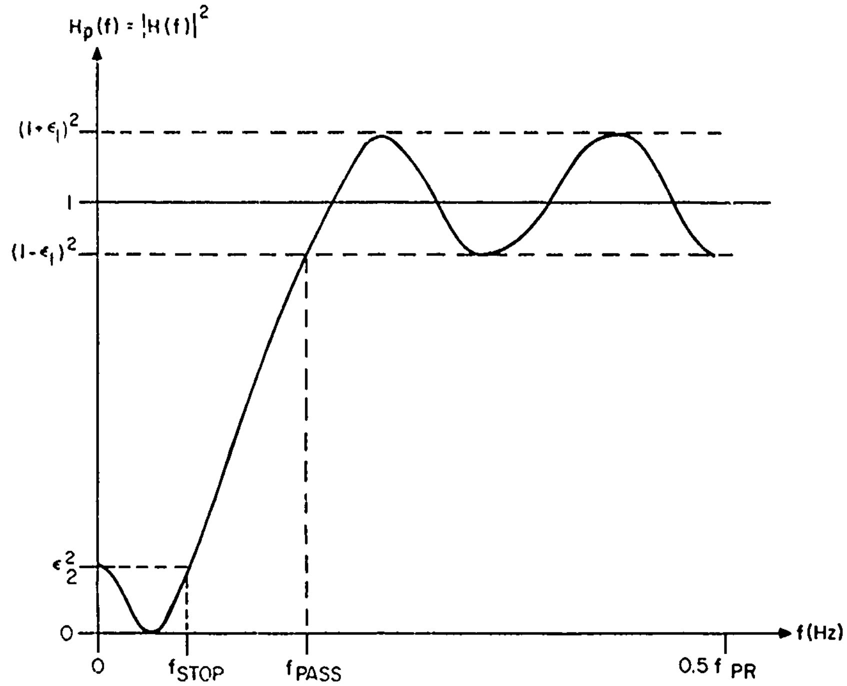

Figure 3(b). Linear-Phase Prototype Filter Frequency Response

$$h_p(n) = K\,h_I(n), \quad -N+1 \leq n \leq N-1 \tag{7}$$

where K is a scaling factor. It is in the determination of the factor K that this development differs from that of [6], for it can be noted that a scaling by the factor $(1+\delta_2)^{-1}$ does not produce a filter $|H(f)|$ with equal (linear) magnitude ripple excursions above and below unity gain.

Since it is the parameters R_{PB} and A_{SB} and thus ε_1 and ε_2 which are specified by the filter designer, it is necessary to invert the above functional relationship to derive expressions for δ_1 and δ_2 in terms of ε_1 and ε_2. This being done, it is a straightforward matter to use the linear-phase FIR filter algorithm to determine the (2N-1)-length IRF $h_o(n)$.

It can be seen from eqn. (6) and Figures 3(b) and 3(c) that

$$(1 + \varepsilon_1)^2 = K(1 + \delta_1 + \delta_2) \tag{8}$$

$$(1 - \varepsilon_1)^2 = K(1 - \delta_1 + \delta_2) \tag{9}$$

and

$$\varepsilon_2^2 = 2K\,\delta_2 \tag{10}$$

Some algebraic manipulation shows that

$$\delta_1 = \frac{4\varepsilon_1}{2 + 2\varepsilon_1^2 - \varepsilon_2^2} \tag{11}$$

$$\delta_2 = \frac{\varepsilon_2^2}{2 + 2\varepsilon_1^2 - \varepsilon_2^2} \tag{12}$$

and

$$K = 1 + \varepsilon_1^2 - \frac{1}{2}\varepsilon_2^2. \tag{13}$$

It is now a simple matter to derive $h_p(n)$ from $h_o(n)$ in terms of these factors as outlined above, and then to derive a nonlinear-phase N-length IRF h(n).

5. GAIN NORMALIZATION

It is often useful to design MTI filters which have 0 dB white-noise gain. This means that the white-noise power in the filtered signal is the same as that in the unfiltered input signal, so that in the absence of clutter, the false-alarm probability P_{FA} is not altered. Such normalization is easily accomplished [3] by invoking Parseval's theorem

$$g^2 = \sum_{n=0}^{n-1} |h(n)|^2 = 2\int_0^{0.5\ f_{PR}} |H(f)|^2\ df \tag{14}$$

to compute the white noise power gain g^2 by means of a simple summation. Scaling h(n) by g^{-1} produces a filter with unity white noise power gain (G_{WNP} = 0 dB) and leaves A_{SB} and R_{PB} unaltered. Note that this procedure is valid for any FIR filter.

It can also be useful to design a filter which has both G_{WNP} = 0 dB and a minimum gain of unity in its passband (G_{PBM} = 0 dB). This means that the probability of detection P_D in the passband will not be reduced by filter attenuation in these passband ripples. This goal can be achieved by an iterative procedure. In this case it is the passband ripple that is altered until the desired characteristics are obtained. The procedure is to specify N, A_{SB} and f_{SB}, select an arbitrary reasonable value for R_{PB} and design a filter using the iterative procedure described above. Based on whether this filter has G_{PBM} greater or less than unity gain, the value for R_{PB} is increased or decreased respectively and another iteration is carried out. This procedure is repeated until convergence to G_{PBM} = 0 dB is achieved to within an acceptable tolerance.

6. COMPUTER PROGRAMS

The program MPMTIPSF generates nonlinear-phase FIR MTI filters for specified values of N, f_{STOP}, f_{PR}, A_{SB} and R_{PB}. Parameters to specify the

grid density for the design of the prototype filter [4] and to select the impulse-response design criterion must also be provided. This program is listed in Appendix A, and the detailed operating instructions are given there. Typical times required to design a filter using a Xerox Sigma 9 computer range from 2.2 sec for N=5 to 28.4 sec for N-16.

The program MPMTIPSFO generates nonlinear-phase FIR MTI filters with G_{PBM} = 0 dB. The same parameters as for the program MPMTIPSF must be specified by the user, but here the parameter R_{PB} is used only as an initial value which is then modified by the program to converge to G_{PBM} = 0 dB. Hence the choice of a starting value for R_{PB} near its final value can significantly reduce the time required to design a filter. This program is listed in Appendix B. Typical running times on the Sigma 9 can be 1 to 10 or more times as long as for the program MPMTIPSF, depending on how good an estimate for the initial value of R_{PB} is used.

Appendix C contains the listings of the subroutines required by the programs MPMTIPSF and MPMTIPSFO. The subroutine REMEZ and its ancillary subroutines have not been included, since they are readily accessible elsewhere [4], but note the modifications required in the dimensions of the COMMON block variables.

7. SUMMARY

It has been shown in Tables 3 and 4 that, for the typically narrow stopbands used in radar MTI filters, nonlinear-phase filters can be designed which offer superior visibility for low-velocity targets, relative to that offered by linear-phase filters designed to the same stopband specifications. It has also been pointed out that the time-domain response of such nonlinear-phase MTI filters can be optimized without altering the filter power response in the frequency domain. The details of the design procedure have been described, and listings of Fortran programs for implementing the procedure have been provided.

8. ACKNOWLEDGEMENT

This work is supported by the Department of National Defence under Research and Development Branch Project No 33C69.

9. REFERENCES

1. Oppenheim, A.V. and R.W. Schafer, *Digital Signal Processing*. Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1975.

2. Houts, R.C. and R.W. Burlage, *Maximizing the Usable Bandwidth of MTI Signal Processors*, IEEE Trans. Aerospace Electronic Systems AES-13, No. 1, January 1977, 48-55.

3. Burlage, D.W. and R.C. Houts, *Design Technique for Improved Bandwidth Moving Target Indicator Processors in Surface Radars*. Report No. RE-TR-75-35, US Army Missile Command, Redstone Arsenal, Alabama, June 1975.

4. McClellan, J.H., T.W. Parks and L.R. Rabiner, *A Computer Program for Designing Optimum FIR Linear Phase Digital Filters*. IEEE Trans. Audio Electroacoust. AU-21, No. 6, December 1973, 506-526.

5. Herrmann, O., *Design of Nonrecursive Digital Filters with Linear Phase*. Electronics Letters 6, No. 11, 28 May 1970, 328-329.

6. Herrmann, O. and W. Schuessler, *Design of Nonrecursive Digital Filters with Minimum Phase*. Electronics Letters 6, No. 11, 28 May 1970.

APPENDIX A

PROGRAM MPMTIPSF

```
C     PROGRAM MPMTIPSF
C
C     WRITTEN BY R.W. HERRING, COMMUNICATIONS RESEARCH CENTRE, OTTAWA
C
C     MODIFIED FROM LINEAR PHASE MTI FILTER DESIGN PROGRAM
C     WRITTEN BY D.W. BURLAGE AND R.C. HOUTS, MICOM.
C
C*********************************************************************
C**   THIS PROGRAM IS USED TO DESIGN MIXED PHASE FIR DIGITAL FILTERS  **
C**   FOR USE IN MTI RADARS AND IS A MODIFIED VERSION OF THE MCCLELLAN**
C**   PROGRAM WHICH EAS PUBLISHED IN THE DECEMBER, 1973 ISSUE OF THE  **
C**   IEEE TRANSACTIONS ON AUDIO AND ELECTROACOUSTICS.                **
C**   MODIFICATIONS INCLUDE:                                          **
C**     1. LIMITED TO HIGHPASS FILTER DESIGN.                         **
C**     2. NORMALIZES COEFFICIENTS W.R.T. SUM H(I)**2  (PGN = 0 DB).  **
C**     3. COEFFICIENTS EQUALIZED FOR OPTIMUM NOISE-REJECTION         **
C**        PERFORMANCE.                                               **
C**   I N P U T   D A T A   C O N S I S T S   O F   O N E   C A R D   **
C**    I N   F R E E   F O R M A T (VARIABLES SEPARTED BY COMMAS) :   **
C**     1. MFILT - NUMBER OF FILTER WEIGHTS (.LE. 75).                **
C**     2. LGRID - GRID DENSITY -- LGRID*MFILT .LE. 1200 .            **
C**                IF LGRID = 0, PROGRAM DEFAULTS TO MAXIMUM VALUE.   **
C**                IF LGRID = 1, PROGRAM USES MINIMUM VALUE FOR LGRID **
C**                WHICH LEADS TO CONVERGENCE.                        **
C**                PROGRAM CHECKS VALUE FOR LGRID AND ENSURES THAT    **
C**                AT LEAST TWO GRID POINTS LIE IN STOP BAND.         **
C**     3. STOPF - UPPER EDGE OF STOPBAND (HZ).                       **
C**     4.  PRF  - PULSE REPETITION FREQUENCY (HZ).                   **
C**     5.  ASB  - STOPBAND ATTENUATION (DB).                         **
C**     6.  RPB  - PASSBAND RIPPLE (DB).                              **
C**     7. MODE  - SELECTS RULE FOR CHOOSING PARTICULAR FILTER        **
C**                IMPULSE RESPONSE:                                  **
C**                IF MODE = 1, MINI-MAX RATIO RULE IS USED.          **
C**                IF MODE = 2, MINIMUM ABSOLUTE SUM RULE IS USED.    **
C**                IF MODE = 3, MINIMUM VARIANCE RULE IS USED.        **
C*********************************************************************
      COMMON /HPFC/ PI,FCU,FUP,WTX,RATIO,ESDEL1,ESDEL2,
     *              NFILT,NEG,NODD,LGRID
      COMMON PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION H(150),ROOTR(150),ROOTI(150)
```

```
      DIMENSION IEXT(79),AD(79),ALPHA(79),X(79),Y(79)
      DIMENSION EXTF(79)
      DIMENSION Z(150)
      DIMENSION DES(1200),GRID(1200),WT(1200)
      DOUBLE PRECISION PI2,PI
      DOUBLE PRECISION AD,DEV,X,Y
      DOUBLE PRECISION H,ROOTR,ROOTI,DTEMP,Z,ZNORM,
     *      DABSUM,SCALE
      LOGICAL SPTFLG
      DATA RATIO/1./,NEG/0/,NODD/1/
      PI = DATAN2(0.D0,-1.D0)
      PI2 = 2.D0*PI
   10 J = NTIMER(0)
      SPTFLG = .FALSE.
C
C     P R O G R A M   I N P U T   S E C T I O N .
C
      INPUT MFILT, LGRID, STOPF, PRF, ASB, RPB , MODE
      IF (MFILT .LE. 1) STOP
      IF (MFILT .LE. 75) GO TO 20
      WRITE (108,15)
   15 FORMAT(/' MFILT SET TO MAX ALLOWED VALUE -- MFILT = 75'/)
      MFILT = 75
C
C     FIND NORMALIZED STOP FREQUENCY, LINEAR RIPPLES AND WEIGHT.
C
   20 FCU = STOPF/PRF
      CRP = 10.**(RPB/20.)
      E1 = (CRP-1.)/(CRP+1.)
      E2 = 10.**(-ASB/20.)*(1.-E1)
      WEIGHT = E1/E2
      DENOM = 2.+(2.*E1*E1-E2*E2)
      D1 = 4.*E1/DENOM
      D2 = E2*E2/DENOM
      WTX = (4.*E1)/(E2*E2)
      NFILT = 2*MFILT-1
      NFCNS = MFILT
      LGRIDM = 1200/NFCNS
C
C     PROGRAM DESIGN MAX 75 TAPS / LGRID = 1200/MFILT
```

```
C
      IF (LGRID .LE. 0) LGRID = LGRIDM
      ITEST = LGRID*NFCNS
C     ENSURE AT LEAST 2 GRID POINTS IN STOPBAND.
      IF (2./FLOAT(ITEST) .LE. FCU) GO TO 30
      LGRID = 2./(FCU*FLOAT(NFCNS))+1.
   25 ITEST = LGRID*NFCNS
   30 IF (ITEST .LE. 1200) GO TO 31
      LGRID = LGRIDM
   31 IF (SPTFLG) GO TO 50
C
C     ESTIMATE PASSBAND FREQUENCY
C
      IF (FCU .GE. 0.04) GO TO 35
C
C     CHEBYSHEV LOWER BOUND ESTIMATE FOR FCU .LT. 0.04
C
      XT = (1.+D1)/D2
      YT = (1.-D1)/D2
      COSHIX = ALOG(XT+SQRT((XT+1.)*(XT-1.)))
      COSHIY = ALOG(YT+SQRT((YT+1.)*(YT-1.)))
      DELF=(COSHIX-SQRT((COSHIX+COSHIY)*(COSHIX-COSHIY)))/(PI*(NFILT-1))
      GO TO 45
C
C     HERRMANN ESTIMATE (BSTJ) FOR FCU .GE. 0.04
C
   35 D1L = ALOG10(D1)
      D2L = ALOG10(D2)
      FK = 11.01217 + 0.51244 * ALOG10(WTX)
      DINF = ((0.005309*D1L+0.07114)*D1L-0.4761)*D2L
     *   -(0.00266*D1L+0.5941)*D1L-0.4278
      DELF = (NFILT-1)*(SQRT(1.+4.*FK*DINF/(NFILT-1)**2)-1.)/(2.*FK)
   45 FUP = FCU+DELF
      IF (FUP .GE. 0.4) FUP = 0.4
      FST = 0.4*DELF
      ITER = 0
      ELAST = 0.
      FLAST = 0.
      DEL = 0.001*D2
      IF (FUP .LE. 0.45) GO TO 50
```

```
       WRITE(108,40) FUP
   40  FORMAT(1HO,'+++UNSUCCESSFUL DESIGN BECAUSE PASSF/PRF = ',F6.4,
     *    ' , WHICH IS GREATER THAN 0.4 +++' )
       GO TO 500
   50  CALL HPFH
       ITER = ITER+1
C
C      CHECK FOR CONVERGENCE
C
       ERROR = D2-ESDEL2
       IF (ABS(ERROR) .LE. DEL) GO TO 70
       FTEMP = FUP
       FUP = FLAST-((FLAST-FUP)*ELAST/(ELAST-ERROR))
       FLAST = FTEMP
       ELAST = ERROR
       IF (FUP) 60,60,50
   60  FUP = FLAST-SIGN(FST,ERROR)
       GO TO 50
   70  PASSF = FUP*PRF
C
C      C A L C U L A T E   I M P U L S E   R E S P O N S E   O F
C              S Q U A R E D   F I L T E R   W I T H
C                  L I N E A R   P H A S E .
C
       NM1 = NFCNS-1
       NZ = NFCNS+1
       DO 305 J = 1,NM1
       H(J) = 0.5*ALPHA(NZ-J)
  305  H(NFILT+1-J) = H(J)
       H(NFCNS) = ALPHA(1)+DEV/WT(1)
C
C      S O L V E   F O R   M I X E D   P H A S E
C        I M P U L S E   R E S P O N S E .
C
       CALL RTSQF(H,NFILT,FCU,ROOTR,ROOTI,MON,MOFF,NCOMB,NBIG)
C
C      EXPAND ROOTS TO DERIVE MAXIMUM PHASE IMPULSE RESPONSE.
C
       DTEMP = DABS(H(1))
       CALL EXPAND(MON+MOFF,ROOTR,ROOT1,MTERMS,Z,H)
```

```
      IF (MTERMS .EQ. MFILT) GO TO 340
      IF (LGRID .EQ. LGRIDM) GO TO 333
C
C     IF WE GET HERE, SPLIT DOUBLE ROOTS FOUND ON UNIT CIRCLE.
C          DOUBLE THE GRID DENSITY AND TRY AGAIN.
C
      LGRID = LGRID*2
      SPTFLG = .TRUE.
      GO TO 25
  333 WRITE (108,335) MFILT,MTERMS,LGRIDM
  335 FORMAT(/' ERROR: MTERMS .NE. MFILT.  MFILT =',I4,',  MTERMS =',
     *      I4,' USING MAXIMUM ALLOWED VALUE FOR LGRID :',I5,'.')
      GO TO 350
C
C     F I N D   T H E   O P T I M U M   M I X E D - P H A S E
C                           F I L T E R .
C
  340 CALL OPTMPF(MODE,MON,MOFF,ROOTR,ROOTI,MTERMS,Z)
C
C     P R O G R A M   O U T P U T   S E C T I O N .
C
  350 WRITE(108,359)
  359 FORMAT(1X)
      CALL NEWPAGE(108)
      WRITE(108,360)
  360 FORMAT(1H1, 82(1H*)//5X,'FINITE IMPULSE RESPONSE (FIR)',
     *   ' OPTIMUM MIXED PHASE DIGITAL FILTER DESIGN ' / 10X,
     *   ' FOR REMOVING GROUND CLUTTER IN MTI RADAR SIGNAL PROCESSOR'//)
      WRITE (108,361) MFILT,LGRID,ITER
  361 FORMAT(23X,I4,' TAP FILTER'
     *      ' (LGRID = ',I4,')'//23X,'CONVERGENCE AFTER',I3,
     *      ' ITERATIONS'/)
      IF (MODE .EQ. 1) WRITE(108,365)
  365 FORMAT(/31X,'MINI-MAX FILTER'/)
      IF (MODE .EQ. 2) WRITE(108,366)
  366 FORMAT(/25X,'MINIMUM ABSOLUTE SUM FILTER'/)
      IF (MODE .EQ. 3) WRITE(108,367)
  367 FORMAT(/23X,'MINIMUM ABSOLUTE VARIANCE FILTER'/)
      SCALE = 2.D0/(DSQRT(1.D0+(FSDEL1+FSDEL2))
     *              +DSQRT(1.D0-(FSDEL1-FSDEL2)))
```

```
      ZNORM = SCALE*DSQRT(DTEMP)
      E1C = 0.5*SCALE*
     *     (DSQRT(1.DO+(ESDEL1+ESDEL2))-DSQRT(1.DO-(ESDEL1-ESDEL2)))
      E2C = SCALE*DSQRT(2.DO*ESDEL2)
      WRITE (108,390)
  390 FORMAT(/31H  BAND  LOWER EDGE  UPPER EDGE  ,5X,
     *        'WEIGHT',5X,'RIPPLE',5X,'RIPPLE(DB)'/2X,66(1H-))
      HPRF = PRF/2.00
      DBSTOP = -20.* ALOG10(E2C/(1.-E1C))
      DBPASS = 20.* ALOG10( (1.+ E1C)/(1.-E1C) )
      WRITE (108,410) STOPF, WEIGHT,  E2C, DBSTOP
  410 FORMAT(2X,'STOP          0.00',2(F12.2),F11.5,F14.2/)
      WRITE (108,420) PASSF, HPRF, E1C, DBPASS
  420 FORMAT (2X,'PASS',2(F12.2),8X,'1.00',F11.5,F14.2
     *     /2X,66(1H-)//)
C
C     NORMALIZE FILTER TO UNITY AMPLITUDE GAIN IN PASSBAND AND
C          CALCULATE NOISE POWER GAIN (PGN) IN DB
C
      PKGN = 20.*ALOG10(1.+E1C)
      SMGN = 20.*ALOG10(1.-E1C)
      SUM = 0.00
      DO 370 K = 1,MFILT
      Z(K) = Z(K)*ZNORM
  370 SUM = SUM+Z(K)*Z(K)
      PGN = 10.00*ALOG10(SUM)
      SRTSUM = SQRT(SUM)
      WRITE (108,373) PGN,PKGN,SMGN,(Z(J),J = 1,MFILT)
  373 FORMAT(1HO,' ORIGINAL TAP GAINS:  NOISE POWER GAIN = ',F7.3,
     * ' DB.'//23X,'MAX GAIN IN PASSBAND ='F7.3,
     * ' DB.'//23X,'MIN GAIN IN PASSBAND ='F7.3,' DB.'/(//2X,10F10.5))
C
C     NORMALIZE Z(K) W.R.T. PGN AND COMPUTE NEW PGN = 0 DB
C
      PKGN = PKGN-PGN
      SMGN = SMGN-PGN
      SUM = 0.
      ZMAX = 0.
      ZMIN = 7.237E75
      DABSUM = 0.DO
```

```
      DO 380 K = 1,MFILT
      Z(K) = Z(K)/SRTSUM
      DTEMP = DABS(Z(K))
      DABSUM = DABSUM+DTEMP
      IF (DTEMP .LE. ZMAX) GO TO 375
      ZMAX = DTEMP
      KMAX = K
  375 IF (DTEMP .GE. ZMIN) GO TO 380
      ZMIN = DTEMP
      KMIN = K
  380 SUM = SUM+Z(K)*Z(K)
      PGN = 10.* ALOG10(SUM)
      WRITE (108,385) PGN,PKGN,SMGN,(Z(J),J = 1,MFILT)
  385 FORMAT(/1H0,' NORMALIZED TAP GAINS:  NOISE POWER GAIN - ',F7.3,
     *   DB.'//25X,'MAX GAIN IN PASSBAND ='F7.3,
     * ' DB.'//25X,'MIN GAIN IN PASSBAND ='F7.3,' DB.'/(///2X,10F10.5))
      TAPRAT = ZMAX/ZMIN
      WRITE (108,387) KMAX,KMIN,TAPRAT,DABSUM,NCOMB,NBIG
  387 FORMAT(/'0 TAP',I3,' HAS GREATEST MAGNITUDE.'
     *       /'0 TAP',I3,' HAS SMALLEST MAGNITUDE.'
     *       /'0 RATIO OF GREATEST TO SMALLEST TAP WEIGHTS IS',F8.2
     *       /'0 ABSOLUTE SUM OF TAP MAGNITUDES IS',F7.3
     *       /'0',I2,' PAIRS OF ROOTS COMBINED.'
     *       /'0',I2,' LARGE ROOTS DISCARDED.'/)
      DO 450 J = 1,NZ
  450 EXTF(J) = GRID(IEXT(J))*PRF
      WRITE(108,455) (EXTF(J),J = 1,NZ)
  455 FORMAT(/' EXTREMAL FREQUENCIES (HZ) '
     *       // (10F10.3/) )
      WRITE(108,460)
  460 FORMAT(1H0,82(1H*))
C
C       O U T P U T   F I L T E R   P A R A M E T E R S   A N D
C             W E I G H T S   F O R   P L O T T I N G .
C
      WRITE(106,480) MFILT,2,LGRID,STOPF,PASSF,PRF,1.,WEIGHT,
     *      DBSTOP,DBPASS,PGN,PKGN,MODE
  480 FORMAT(I3,I1,I4,2F8.3,F5.0,F5.3,5F8.3,I1)
      WRITE(106,490) (Z(J),J = 1,MFILT)
  490 FORMAT(10F8.5)
```

```
  500 CONTINUE
      J = NTIMER(1)
      TEMP = FLOAT(J)/500.
      WRITE (108,510) TEMP
  510 FORMAT(// COMPUTATION REQUIRED',F7.2,' SECONDS OF CPU TIME.'/)
      GO TO 10
      END
```

APPENDIX B

PROGRAM MPMTIPSFO

```
C     PROGRAM MFMTIPSFO
C
C     WRITTEN BY R.W. HERRING, COMMUNICATIONS RESEARCH CENTRE, OTTAWA
C
C     MODIFIED FROM LINEAR PHASE MTI FILTER DESIGN PROGRAM
C     WRITTEN BY D.W. BURLAGE AND R.C. HOUTS, MICOM.
C
C**************************************************************************
C**   THIS PROGRAM IS USED TO DESIGN MIXED PHASE FIR DIGITAL FILTERS  **
C**   FOR USE IN MTI RADARS AND IS A MODIFIED VERSION OF THE MCCLELLAN**
C**   PROGRAM WHICH EAS PUBLISHED IN THE DECEMBER, 1973 ISSUE OF THE  **
C**   IEEE TRANSACTIONS ON AUDIO AND ELECTROACOUSTICS.                **
C**   MODIFICATIONS INCLUDE:                                          **
C**     1. LIMITED TO HIGHPASS FILTER DESIGN.                         **
C**     2. NORMALIZES COEFFICIENTS W.R.T. SUM H(I)**2  (PGN = 0 DB).  **
C**     3. MINIMUM GAIN OF 0 DB IN PASSBAND.                          **
C**     4. COEFFICIENTS EQUALIZED FOR OPTIMUM NOISE-REJECTION         **
C**        PERFORMANCE.                                               **
C**   I N P U T   D A T A   C O N S I S T S   O F   O N E   C A R D   **
C**    I N   F R E E   F O R M A T (VARIABLES SEPARTED BY COMMAS) :   **
C**     1. MFILT - NUMBER OF FILTER WEIGHTS (.LE. 75).                **
C**     2. LGRID - GRID DENSITY -- LGRID*MFILT .LE. 1200 .            **
C**                IF LGRID = 0, PROGRAM DEFAULTS TO MAXIMUM VALUE.   **
C**                IF LGRID = 1, PROGRAM USES MINIMUM VALUE FOR LGRID **
C**                WHICH LEADS TO CONVERGENCE.                        **
C**                PROGRAM CHECKS VALUE FOR LGRID AND ENSURES THAT    **
C**                AT LEAST TWO GRID POINTS LIE IN STOP BAND.         **
C**     3. STOPF - UPPER EDGE OF STOPBAND (HZ).                       **
C**     4.  PRF  - PULSE REPETITION FREQUENCY (HZ).                   **
C**     5.  ASB  - STOPBAND ATTENUATION (DB).                         **
C**     6.  RPB  - PASSBAND RIPPLE (DB).                              **
C**     7. MODE  - SELECTS RULE FOR CHOOSING PARTICULAR FILTER        **
C**                IMPULSE RESPONSE:                                  **
C**                IF MODE = 1, MINI-MAX RATIO RULE IS USED.          **
C**                IF MODE = 2, MINIMUM ABSOLUTE SUM RULE IS USED.    **
C**                IF MODE = 3, MINIMUM VARIANCE RULE IS USED.        **
C**************************************************************************
      COMMON /HPFC/ PI,FCU,FUP,WTX,RATIO,ESDEL1,ESDEL2,
     *              NFILT,NEG,NODD,LGRID
      COMMON PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
```

```
      DIMENSION H(150),ROOTR(150),ROOTI(150)
      DIMENSION IEXT(79),AD(79),ALPHA(79),X(79),Y(79)
      DIMENSION EXTF(79)
      DIMENSION Z(150)
      DIMENSION DES(1200),GRID(1200),WT(1200)
      DOUBLE PRECISION PI2,PI
      DOUBLE PRECISION AD,DEV,X,Y
      DOUBLE PRECISION H,ROOTR,ROOTI,DTEMP,Z,ZNORM,
     *     DABSUM,SCALE
      LOGICAL SPTFLG
      DATA RATIO/1./,NEG/0/,NODD/1/
      PI = DATAN2(0.D0,-1.D0)
      PI2 = 2.D0*PI
   10 J = NTIMER(0)
      SPTFLG = .FALSE.
      ITER = 0
C
C     P R O G R A M   I N P U T   S E C T I O N .
C
      INPUT MFILT, LGRID, STOPF, PRF, ASB, RPB , MODE
      IF (MFILT .LE. 1) STOP
      IF (MFILT .LE. 75) GO TO 20
      WRITE (108,15)
   15 FORMAT(/' MFILT SET TO MAX ALLOWED VALUE -- MFILT = 75'/)
      MFILT = 75
C
C     FIND NORMALIZED STOP FREQUENCY, LINEAR RIPPLES AND WEIGHT.
C
   20 FCU = STOPF/PRF
      CRP = 10.**(RPB/20.)
      E1 = (CRP-1.)/(CRP+1.)
      E2 = 10.**(-ASB/20.)*(1.-E1)
      WEIGHT = E1/E2
      DENOM = 2.+(2.*E1*E1-E2*E2)
      D1 = 4.*E1/DENOM
      D2 = E2*E2/DENOM
      WTX = (4.*E1)/(E2*E2)
      IF (ITER .NE. 0) GO TO 50
      NFILT = 2*MFILT-1
      NFCNS = MFILT
```

```
        LGRIDM = 1200/NFCNS
C
C       PROGRAM DESIGN MAX 75 TAPS / LGRID = 1200/MFILT
C
        IF (LGRID .LE. 0) LGRID = LGRIDM
        ITEST = LGRID*NFCNS
C       ENSURE AT LEAST 2 GRID POINTS IN STOPBAND.
        IF (2./FLOAT(ITEST) .LE. FCU) GO TO 30
        LGRID = 2./(FCU*FLOAT(NFCNS))+1.
   25   ITEST = LGRID*NFCNS
   30   IF (ITEST .LE. 1200) GO TO 31
        LGRID = LGRIDM
   31   IF (SPTFLG) GO TO 50
C
C       ESTIMATE PASSBAND FREQUENCY
C
        IF (FCU .GE. 0.04) GO TO 35
C
C       CHEBYSHEV LOWER BOUND ESTIMATE FOR FCU .LT. 0.04
C
        XT = (1.+D1)/D2
        YT = (1.-D1)/D2
        COSHIX = ALOG(XT+SQRT((XT+1.)*(XT-1.)))
        COSHIY = ALOG(YT+SQRT((YT+1.)*(YT-1.)))
        DELF=(COSHIX-SQRT((COSHIX+COSHIY)*(COSHIX-COSHIY)))/(PI*(NFILT-1))
        GO TO 45
C
C       HERRMANN ESTIMATE (BSTJ) FOR FCU .GE. 0.04
C
   35   D1L = ALOG10(D1)
        D2L = ALOG10(D2)
        FK = 11.01217 + 0.51244 * ALOG10(WTX)
        DINF = ((0.005309*D1L+0.07114)*D1L-0.4761)*D2L
       *   -(0.00266*D1L+0.5941)*D1L-0.4278
        DELF = (NFILT-1)*(SQRT(1.+4.*FK*DINF/(NFILT-1)**2)-1.)/(2.*FK)
   45   FUP = FCU+DELF
        IF (FUP .GE. 0.4) FUP = 0.4
        FST = 0.4*DELF
        ELAST = 0.
        FLAST = 0.
```

```
      DEL = 0.0001*D2
   49 IF (FUP .LE. 0.45) GO TO 50
      WRITE(108,40) FUP
   40 FORMAT(1H0,'+++UNSUCCESSFUL DESIGN BECAUSE PASSF/PRF = ',F6.4,
     *   ' , WHICH IS GREATER THAN 0.4 +++' )
      GO TO 500
   50 CALL HPFH
      ITER = ITER+1
C
C     CHECK FOR CONVERGENCE
C
      ERROR = D2-ESDEL2
      IF (ABS(ERROR) .LE. DEL) GO TO 300
      FTEMP = FUP
      FUP = FLAST-((FLAST-FUP)*ELAST/(ELAST-ERROR))
      FLAST = FTEMP
      ELAST = ERROR
      IF (FUP) 60,60,49
   60 FUP = FLAST-SIGN(FST,ERROR)
      GO TO 50
C
C     C A L C U L A T E   I M P U L S E   R E S P O N S E   O F
C             S Q U A R E D   F I L T E R   W I T H
C                   L I N E A R   P H A S E .
C
  300 NM1 = NFCNS-1
      NZ = NFCNS+1
      DO 305 J = 1,NM1
      H(J) = 0.5*ALPHA(NZ-J)
  305 H(NFILT+1-J) = H(J)
      H(NFCNS) = ALPHA(1)+DEV/WT(1)
C
C     S O L V E   F O R   M I X E D   P H A S E
C        I M P U L S E   R E S P O N S E .
C
      CALL RTSQF(H,NFILT,FCU,ROOTR,ROOTI,MON,MOFF,NCOMB,NBIG)
C
C     EXPAND ROOTS TO DERIVE MAXIMUM PHASE IMPULSE RESPONSE.
C
      DTEMP = DABS(H(1))
```

```
      CALL EXPAND(MON+MOFF,ROOTR,ROOTI,MTERMS,Z,H)
      IF (MTERMS .EQ. MFILT) GO TO 340
      IF (LGRID .EQ. LGRIDM) GO TO 333
C
C     IF WE GET HERE, SPLIT DOUBLE ROOTS FOUND ON UNIT CIRCLE.
C          DOUBLE THE GRID DENSITY AND TRY AGAIN.
C
      LGRID = LGRID*2
      SPTFLG = .TRUE.
      GO TO 25
  333 WRITE (108,335) MFILT,MTERMS,LGRIDM
  335 FORMAT(/' ERROR: MTERMS .NE. MFILT.  MFILT =',I4,',  MTERMS =',
     *     I4,' USING MAXIMUM ALLOWED VALUE FOR LGRID :',I5,'.')
      GO TO 350
C
C     F I N D   T H E   O P T I M U M   M I X E D - P H A S E
C                        F I L T E R .
C
  340 CALL OPTMPF(MODE,MON,MOFF,ROOTR,ROOTI,MTERMS,Z)
C
C     P R O G R A M   O U T P U T   S E C T I O N .
C
  350 SCALE = 2.D0/(DSQRT(1.D0+(ESDEL1+ESDEL2))
     *              +DSQRT(1.D0-(ESDEL1-ESDEL2)))
      ZNORM = SCALE*DSQRT(DTEMP)
      E1C = 0.5*SCALE*
     *     (DSQRT(1.D0+(ESDEL1+ESDEL2))-DSQRT(1.D0-(ESDEL1-ESDEL2)))
      E2C = SCALE*DSQRT(2.D0*ESDEL2)
      PASSF = FUP*PRF
      HPRF = PRF/2.00
      DBSTOP = -20.* ALOG10(E2C/(1.-E1C))
      DBPASS = 20.* ALOG10( (1.+ E1C)/(1.-E1C) )
C
C     NORMALIZE FILTER TO UNITY AMPLITUDE GAIN IN PASSBAND AND
C          CALCULATE NOISE POWER GAIN (PGN) IN DB
C
      PKGN = 20.*ALOG10(1.+E1C)
      SMGN = 20.*ALOG10(1.-E1C)
      SUM = 0.00
      DO 370 K = 1,MFILT
```

```
      Z(K) = Z(K)*ZNORM
  370 SUM = SUM+Z(K)*Z(K)
      PGN = 10.00*ALOG10(SUM)
      SRTSUM = SQRT(SUM)
      IF (ABS(SMGN-PGN) .LE. 0.001) GO TO 358
      RPB = PKGN-PGN
      GO TO 20
  358 WRITE(108,359)
  359 FORMAT(1X)
      CALL NEWPAGE(108)
      WRITE(108,360)
  360 FORMAT(1H1, 82(1H*)//5X,'FINITE IMPULSE RESPONSE (FIR)',
     *   ' OPTIMUM MIXED PHASE DIGITAL FILTER DESIGN ' / 10X,
     *   ' FOR REMOVING GROUND CLUTTER IN MTI RADAR SIGNAL PROCESSOR'//)
      WRITE (108,361) MFILT,LGRID,ITER
  361 FORMAT(23X,I4,' TAP FILTER'
     *     ' (LGRID = ',I4,')'//23X,'CONVERGENCE AFTER',I3,
     *     ' ITERATIONS'/)
      IF (MODE .EQ. 1) WRITE(108,365)
  365 FORMAT(/31X,'MINI-MAX FILTER'/)
      IF (MODE .EQ. 2) WRITE(108,366)
  366 FORMAT(/25X,'MINIMUM ABSOLUTE SUM FILTER'/)
      IF (MODE .EQ. 3) WRITE(108,367)
  367 FORMAT(/23X,'MINIMUM ABSOLUTE VARIANCE FILTER'/)
      WRITE (108,390)
  390 FORMAT(/31H  BAND  LOWER EDGE  UPPER EDGE  ,5X,
     *        'WEIGHT',5X,'RIPPLE',5X,'RIPPLE(DB)'/2X,66(1H-))
      WRITE (108,410) STOPF, WEIGHT,  E2C, DBSTOP
  410 FORMAT(2X,'STOP         0.00',2(F12.2),F11.5,F14.2/)
      WRITE (108,420) PASSF, HPRF, E1C, DBPASS
  420 FORMAT (2X,'PASS',2(F12.2),8X,'1.00',F11.5,F14.2
     *      /2X,66(1H-)//)
      WRITE (108,373) PGN,PKGN,SMGN,(Z(J),J = 1,MFILT)
  373 FORMAT(1H0,' ORIGINAL TAP GAINS:  NOISE POWER GAIN = ',F7.3,
     * ' DB.'//23X,'MAX GAIN IN PASSBAND ='F7.3,
     * ' DB.'//23X,'MIN GAIN IN PASSBAND ='F7.3,' DB.'/(//2X,10F10.5))
C
C     NORMALIZE Z(K) W.R.T  PGN AND COMPUTE NEW PGN = 0 DB
C
      PKGN = PKGN-PGN
```

```
      SMGN = SMGN-PGN
      SUM = 0.
      ZMAX = 0.
      ZMIN = 7.237E75
      DABSUM = 0.D0
      DO 380 K = 1,MFILT
      Z(K) = Z(K)/SRTSUM
      DTEMP = DABS(Z(K))
      DABSUM = DABSUM+DTEMP
      IF (DTEMP .LE. ZMAX) GO TO 375
      ZMAX = DTEMP
      KMAX = K
  375 IF (DTEMP .GE. ZMIN) GO TO 380
      ZMIN = DTEMP
      KMIN = K
  380 SUM = SUM+Z(K)*Z(K)
      PGN = 10.* ALOG10(SUM)
      WRITE (108,385) PGN,PKGN,SMGN,(Z(J),J = 1,MFILT)
  385 FORMAT(/1H0,' NORMALIZED TAP GAINS:  NOISE POWER GAIN = ',F7.3,
     * ' DB.'//25X,'MAX GAIN IN PASSBAND ='F7.3,
     * ' DB.'//25X,'MIN GAIN IN PASSBAND ='F7.3,' DB.'/(//2X,10F10.5))
      TAPRAT = ZMAX/ZMIN
      WRITE (108,387) KMAX,KMIN,TAPRAT,DABSUM,NCOMB,NBIG
  387 FORMAT(/'0 TAP',I3,' HAS GREATEST MAGNITUDE.'
     *       /'0 TAP',I3,' HAS SMALLEST MAGNITUDE.'
     *       /'0 RATIO OF GREATEST TO SMALLEST TAP WEIGHTS IS',F8.2
     *       /'0 ABSOLUTE SUM OF TAP MAGNITUDES IS',F7.3
     *       /'0',I2,' PAIRS OF ROOTS COMBINED.'
     *       /'0',I2,' LARGE ROOTS DISCARDED.'/)
      DO 450 J = 1,NZ
  450 EXTF(J) = GRID(IEXT(J))*PRF
      WRITE(108,455) (EXTF(J),J = 1,NZ)
  455 FORMAT(/'  EXTREMAL FREQUENCIES (HZ) '
     *        // (10F10.3/) )
      WRITE(108,460)
  460 FORMAT(1H0,82(1H*))
C
C     O U T P U T   F I L T E R   P A R A M E T E R S   A N D
C          W E I G H T S   F O R   P L O T T I N G .
C
```

```
      WRITE(106,480) MFILT,2,LGRID,STOPF,PASSF,PRF,1.,WEIGHT,
     *       DBSTOP,DBPASS,FGN,FKGN,MODE
  480 FORMAT(I3,I1,I4,2F8.3,F5.0,F5.3,5F8.3,I1)
      WRITE(106,490) (Z(J),J = 1,MFILT)
  490 FORMAT(10F8.5)
  500 CONTINUE
      J = NTIMER(1)
      TEMP = FLOAT(J)/500.
      WRITE (108,510) TEMP
  510 FORMAT(// COMPUTATION REQUIRED',F7.2,' SECONDS OF CPU TIME.'/)
      GO TO 10
      END
```

APPENDIX C

SUBROUTINES

```
      SUBROUTINE HPFH
      COMMON /HPFC/ PI,STOPF,PASSF,WEIGHT,RATIO,ESDEL1,ESDEL2,
     *              NFILT,NEG,NODD,LGRID
      COMMON PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION IEXT(79),AD(79),ALPHA(79),X(79),Y(79),EDGE(4),
     *          DES(1200),GRID(1200),WT(1200)
      DOUBLE PRECISION PI2,PI
      DOUBLE PRECISION AD,DEV,X,Y
      DATA EDGE(1)/0./,EDGE(4)/0.5/
      EDGE(2) = STOPF
      EDGE(3) = PASSF
C**   FIND THE DESIRED MAGNITUDE (DES(J)) AND WEIGHT (WT(J)) ON GRID.
  140 GRID(1) = EDGE(1)
      DELF = 0.5 / FLOAT(LGRID * NFCNS)
      J=1
      FUP=EDGF(2)
      IF(NEG.EQ.0) GO TO 145
      GRID(1) = DELF
  145 TEMP=GRID(J)
      DES(J) = 0.
      WT(J) = WEIGHT * (1.00-(1.00-RATIO)*TEMP/EDGE(2))
      J=J+1
      GRID(J)=TEMP+DELF
      IF(GRID(J).GT.FUP) GO TO 150
      GO TO 145
  150 GRID(J-1)=FUP
      WT(J-1)= RATIO * WEIGHT
      GRID(J)= EDGE(3)
      FUP= 0.50
  155 TEMP= GRID(J)
      DES(J)= 1.00
      WT(J)= 1.
      J= J + 1
      GRID(J)= TEMP + DELF
      IF (GRID(J).GT.0.50) GO TO 160
      GO TO 155
  160 GRID(J-1)= 0.50
      NGRID= J-1
C** SET UP APPROXIMATION PROBLEM.  WEIGHT BY SIN(PI*GRID(J)) IF NFILT EVEN
      IF (NEG.EQ.1) GO TO 175
```

```
      IF(NODD.EQ.1) GO TO 185
C  NFILT EVEN; SYMMETRY POS.
      IF(GRID(NGRID).GT.(0.5-DELF)) NGRID=NGRID-1
      DO 170 J=1,NGRID
      CHANGE=DCOS(PI*GRID(J))
      DES(J)=DES(J)/CHANGE
  170 WT(J)=WT(J)*CHANGE
      GO TO 185
C  NFILT EVEN; SYMMETRY NEG.
 175  DO 176 J=1,NGRID
      CHANGE=DSIN(PI*GRID(J))
      DES(J)= DES(J) / CHANGE
  176 WT(J)= WT(J) * CHANGE
C  NFILT ODD; SYMMETRY POS.
C INITIAL GUESS FOR EXTREMAL FREQUENCIES IS EQUALLY SPACED ON GRID
 185  TEMP=FLOAT(NGRID-1)/FLOAT(NFCNS)
      DO 190 J=1,NFCNS
 190  IEXT(J)=(J-1)*TEMP+1
      IEXT(NFCNS+1)=NGRID
      CALL REMEZ(EDGE,2)
      ESDEL1 = DEV
 200  ESDEL2 = DEV / WEIGHT
      RETURN
      END
```

```
      SUBROUTINE RTSQF(H,NFILT,FCU,ROOTR,ROOTI,MON,MOFF,NCOMB,NBIG)
C
C     HERRING  28 JULY, 1977.
C
C     SUBROUTINE TO FIND AND SORT ROOTS OF SQUARED FILTER
C          RESPONSE IN DESIGN OF MIXED-PHASE FILTERS.
C
      COMMON PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION H(150),ROOTR(150),ROOTI(150)
      DIMENSION IEXT(79),AD(79),ALPHA(79),X(79),Y(79)
      DIMENSION DES(1200),GRID(1200),WT(1200)
      DOUBLE PRECISION PI2
      DOUBLE PRECISION AD,DEV,X,Y
      DOUBLE PRECISION H,ROOTR,ROOTI,DTEMP,PSPLIT
C
      NCOMB = 0
      NBIG = 0
C
C     FIND ROOTS OF SQUARED FILTER IMPULSE RESPONSE.
C
      NROOTS = NFILT-1
      CALL MULPOL(H,NROOTS,ROOTR,ROOTI)
C
C     CONVERT ROOTS TO POLAR FORM AND RETAIN ONLY THOSE ROOTS
C          ON UPPER HALF OF Z-PLANE AND ON OR OUTSIDE THE
C          UNIT CIRCLE.
C
      NREAL = 0
      K = 0
      DO 320 J = 1,NROOTS
C     ELIMINATE SPLITTING OF ROOTS NEAR REAL AXIS.
      IF (DABS(ROOTI(J)) .GT. 2.D-5) GO TO 300
      ROOTI(J) = 0.D0
C     COUNT AS REAL ROOT IF IT LIES ON OR OUTSIDE UNIT CIRCLE.
      IF (DABS(ROOTR(J)) .GE. 0.99998D0) NREAL = NREAL+1
  300 DTEMP = ROOTR(J)*ROOTR(J)+ROOTI(J)*ROOTI(J)
C     MOVE NEARBY ROOTS ONTO UNIT CIRCLE.
      IF (DABS(DTEMP-1.D0) .GT. 4.D-5) GO TO 310
      DTEMP = 1.D0
  310 ROOTI(J) = DATAN2(ROOTI(J),ROOTR(J))
```

```
      ROOTR(J) = DTEMP
      IF (ROOTI(J) .LT. 0.D0) GO TO 320
      IF (ROOTR(J) .LT. 1.D0) GO TO 320
      K = K+1
      ROOTR(K) = DSQRT(ROOTR(J))
      ROOTI(K) = ROOTI(J)
  320 CONTINUE
C
C     SORT ROOTS IN ORDER OF DESCENDING MAGNITUDE.
C
      JROOTS = K
      DO 330 J = 2,JROOTS
      KHI = JROOTS+1-J
      DO 330 K = 1,KHI
      IF (ROOTR(K) .GE. ROOTR(K+1)) GO TO 330
      DTEMP = ROOTR(K)
      ROOTR(K) = ROOTR(K+1)
      ROOTR(K+1) = DTEMP
      DTEMP = ROOTI(K)
      ROOTI(K) = ROOTI(K+1)
      ROOTI(K+1) = DTEMP
  330 CONTINUE
C
C     COUNT ROOTS ON UNIT CIRCLE.
C
      DO 340 J = 1,JROOTS
      J1 = JROOTS+1-J
      IF (ROOTR(J1) .NE. 1.D0) GO TO 350
  340 CONTINUE
  350 J1 = JROOTS-J1
      IF (J1 .LE. 1) GO TO 400
C
C     SORT ROOTS ON UNIT CIRCLE IN ORDER OF
C           ASCENDING PHASE.
C
      KLO = JROOTS+1-J1
      DO 360 J = 2,J1
      KHI = JROOTS+1-J
      DO 360 K = KLO,KHI
      IF (ROOTI(K) .LT. ROOTI(K+1)) GO TO 360
```

```
      ROOTR(J) = DTEMP
      IF (ROOTI(J) .LT. 0.D0) GO TO 320
      IF (ROOTR(J) .LT. 1.D0) GO TO 320
      K = K+1
      ROOTR(K) = DSQRT(ROOTR(J))
      ROOTI(K) = ROOTI(J)
  320 CONTINUE
C
C     SORT ROOTS IN ORDER OF DESCENDING MAGNITUDE.
C
      JROOTS = K
      DO 330 J = 2,JROOTS
      KHI = JROOTS+1-J
      DO 330 K = 1,KHI
      IF (ROOTR(K) .GE. ROOTR(K+1)) GO TO 330
      DTEMP = ROOTR(K)
      ROOTR(K) = ROOTR(K+1)
      ROOTR(K+1) = DTEMP
      DTEMP = ROOTI(K)
      ROOTI(K) = ROOTI(K+1)
      ROOTI(K+1) = DTEMP
  330 CONTINUE
C
C     COUNT ROOTS ON UNIT CIRCLE.
C
      DO 340 J = 1,JROOTS
      J1 = JROOTS+1-J
      IF (ROOTR(J1) .NE. 1.D0) GO TO 350
  340 CONTINUE
  350 J1 = JROOTS-J1
      IF (J1 .LE. 1) GO TO 400
C
C     SORT ROOTS ON UNIT CIRCLE IN ORDER OF
C           ASCENDING PHASE.
C
      KLO = JROOTS+1-J1
      DO 360 J = 2,J1
      KHI = JROOTS+1-J
      DO 360 K = KLO,KHI
      IF (ROOTI(K) .LT. ROOTI(K+1)) GO TO 360
```

```
      DTEMP = ROOTR(K)
      ROOTR(K) = ROOTR(K+1)
      ROOTR(K+1) = DTEMP
      DTEMP = ROOTI(K)
      ROOTI(K) = ROOTI(K+1)
      ROOTI(K+1) = DTEMP
  360 CONTINUE
C
C     ELIMINATE SPLITTING OF DOUBLE ROOTS ON UNIT CIRCLE.
C
      PSPLIT = PI2*FCU/FLOAT(2*J1+1)
      K = KLO
      KHI = JROOTS-1
      J = 0
  370 IF (ROOTI(K+1)-ROOTI(K) .GE. PSPLIT) GO TO 380
      ROOTI(KLO+J) = 0.5D0*(ROOTI(K)+ROOTI(K+1))
      IF (ROOTI(KLO+J) .EQ. 0.D0) NREAL = NREAL-1
      J = J+1
      K = K+1
  380 K = K+1
      IF (K .LE. KHI) GO TO 370
      ROOTI(KLO+J) = ROOTI(K)
      JROOTS = JROOTS-J
      J1 = J1-J
      NCOMB = J
C
C     IF CORRECT ORDER OF FILTER CAN BE DESIGNED, RETURN.
C
      IF (2*(JROOTS-NREAL)+NREAL .EQ. NROOTS/2) GO TO 400
C
C     OTHERWISE, DISCARD LARGEST ROOT.
C
      DO 390 J = 2,JROOTS
      ROOTR(J-1) = ROOTR(J)
      ROOTI(J-1) - ROOTI(J)
  390 CONTINUE
      JROOTS = JROOTS-1
      NBIG = 1
  400 MON = J1
      MOFF = JROOTS-J1
```

```
      RETURN
      END
```

```
      SUBROUTINF MULPOL(COE,N1,ROOTR,ROOTI)
C
C     SUBROUTINE MULPOL FACTORS A POLYNOMIAL BY MULLER'S ALGORITHM
C
C     SOURCE : D.S. HUMPHERYS, 'THE ANALYSIS, DESIGN AND SYNTHESIS
C                 OF ELECTRICAL FILTERS', PP 649-652.
C                 PRENTICE-HALL, 1970.
C
      DIMENSION COE(1),ROOTR(1),ROOTI(1)
C
C     FOR DOUBLE PRECISION VERSION REMOVE THE C IN COLUMN 1
C          OF THE FOLLOWING LINE:
      DOUBLE PRECISION COE,ROOTR,ROOTI
C
      DOUBLE PRECISION ALP1R,ALP1I,ALP2R,ALP2I,ALP3R,ALP3I,ALP4R,ALP4I,
     *     AXR,AXI,HELL,BELL,BET1R,BET1I,BET2R,BET2I,BET3R,BET3I,
     *     DE15,DE16,TEM,TEMR,TEMI,TEM1,TEM2,TE1,TE2,TE3,TE4,TE5,TE6,
     *     TE7,TE8,TE9,TE10,TE11,TE12,TE13,TE14,TE15,TE16
      N2 = N1+1
      N4 = 0
      I = N1+1
   19 IF (COE(I)) 9,7,9
    7 N4 = N4+1
      ROOTR(N4) = 0.D0
      ROOTI(N4) = 0.D0
      I = I-1
      IF (N4-N1) 19,37,19
    9 CONTINUE
   10 AXR = 0.8D0
      AXI = 0.0D0
      L = 1
      N3 = 1
      ALP1R = AXR
      ALP1I = AXI
      M = 1
      GO TO 99
   11 BET1R = TEMR
      BET1I = TEMI
      AXR = 0.85D0
      ALP2R = AXR
```

```
      ALP2I = AXI
      M = 2
      GO TO 99
12    BET2R = TEMR
      BET2I = TEMI
      AXR = 0.9D0
      ALP3R = AXR
      ALP3I = AXI
      M = 3
      GO TO 99
13    BET3R = TEMR
      BET3I = TEMI
14    TE1 = ALP1R-ALP3R
      TE2 = ALP1I-ALP3I
      TE5 = ALP3R-ALP2R
      TE6 = ALP3I-ALP2I
      TEM = TE5*TE5+TE6*TE6
      TE3 = (TE1*TE5+TE2*TE6)/TEM
      TE4 = (TE2*TE5-TE1*TE6)/TEM
      TE7 = TE3+1.0D0
      TE9 = TE3*TE3-TE4*TE4
      TE10 = 2.D0*TE3*TE4
      DE15 = TE7*BET3R-TE4*BET3I
      DE16 = TE7*BET3I+TE4*BET3R
      TE11 = TE3*BET2R-TE4*BET2I+BET1R-DE15
      TE12 = TE3*BET2I+TE4*BET2R+BET1I-DE16
      TE7 = TE9-1.0D0
      TE1 = TE9*BET2R-TE10*BET2I
      TE2 = TE9*BET2I+TE10*BET2R
      TE13 = TE1-BET1R-TE7*BET3R+TE10*BET3I
      TE14 = TE2-BET1I-TE7*BET3I-TE10*BET3R
      TE15 = DE15*TE3-DE16*TE4
      TE16 = DE15*TE4+DE16*TE3
      TE1 = TE13*TE13-TE14*TE14-4.D0*(TE11*TE15-TE12*TE16)
      TE2 = 2.D0*TE13*TE14-4.D0*(TE12*TE15+TE11*TE16)
      TEM = DSQRT(TE1*TE1+TE2*TE2)
      IF (TE1) 113,113,112
113   TE4 = DSQRT(.5D0*(TEM-TE1))
      TE3 = .5D0*TE2/TE4
      GO TO 111
```

```
112   TE3 = DSQRT(.5D0*(TEM+TE1))
      IF (TE2) 110,200,200
110   TE3 = -TE3
200   TE4 = .5D0*TE2/TE3
111   TE7 = TE13+TE3
      TE8 = TE14+TE4
      TE9 = TE13-TE3
      TE10 = TE14-TE4
      TE1 = 2.D0*TE15
      TE2 = 2.D0*TE16
      IF (TE7*TE7+TE8*TE8-TE9*TE9-TE10*TE10) 204,204,205
204   TE7 = TE9
      TE8 = TE10
205   TEM = TE7*TE7+TE8*TE8
      TE3 = (TE1*TE7+TE2*TE8)/TEM
      TE4 = (TE2*TE7-TE1*TE8)/TEM
      AXR = ALP3R+TE3*TE5-TE4*TE6
      AXI = ALP3I+TE3*TE6+TE4*TE5
      ALP4R = AXR
      ALP4I = AXI
      M = 4
      GO TO 99
15    N6 = 1
38    IF (DABS(HELL)+DABS(BELL)-1.D-20) 18,18,16
16    TE7 = DABS(ALP3R-AXR)+DABS(ALP3I-AXI)
      IF (TE7/(DABS(AXR)+DABS(AXI))-1.D-12) 18,18,17
17    N3 = N3+1
      ALP1R = ALP2R
      ALP1I = ALP2I
      ALP2R = ALP3R
      ALP2I = ALP3I
      ALP3R = ALP4R
      ALP3I = ALP4I
      BET1R = BET2R
      BET1I = BET2I
      BET2R = BET3R
      BET2I = BET3I
      BET3R = TEMR
      BET3I = TEMI
      IF (N3-100) 14,18,18
```

```
18    N4 = N4+1
      ROOTR(N4) = ALP4R
      ROOTI(N4) = ALP4I
      N3 = 0
41    IF (N4-N1) 30,37,37
37    RETURN
30    IF (DABS(ROOTI(N4))-1.D-5) 10,10,31
31    GO TO (32,10),L
32    AXR = ALP1R
      AXI = -ALP1I
      ALP1I = -ALP1I
      M = 5
      GO TO 99
33    BET1R = TEMR
      BET1I = TEMI
      AXR = ALP2R
      AXI = -ALP2I
      ALP2I = -ALP2I
      M = 6
      GO TO 99
34    BET2R = TEMR
      BET2I = TEMI
      AXR = ALP3R
      AXI = -ALP3I
      ALP3I = -ALP3I
      L = 2
      M = 3
99    TEMR = COE(1)
      TEMI = 0.D0
      DO 100 I = 1,N1
      TE1 = TEMR*AXR-TEMI*AXI
      TEMI = TEMI*AXR+TEMR*AXI
100   TEMR = TE1+COE(I+1)
      HELL = TEMR
      BELL = TEMI
42    IF (N4) 102,103,102
102   DO 101 I = 1,N4
      TEM1 = AXR-ROOTR(I)
      TEM2 = AXI-ROOTI(I)
      TE1 = TEM1*TEM1+TEM2*TEM2
```

```
      TE2 = (TEMR*TEM1+TEMI*TEM2)/TE1
      TEMI = (TEMI*TEM1-TEMR*TEM2)/TE1
101   TEMR = TE2
103   GO TO (11,12,13,15,33,34),M
      END
```

```
      SUBROUTINE OPTMPF(MODE,NON,NOFF,ROOTM,ROOTP,NTERMS,Z)
C
C     HERRING  18 FEB, 1977.  REVISED 05 AUG, 1977.
C
      COMMON PI2,AD(79),DEV,X(79),Y(79),GRID(1200),ROOTR(75),ROOTT(75),
     *     ROOTI(75),ZON(75),ZOFF(75),ZTEMP(75),WORK(150)
      DOUBLE PRECISION PI2,AD,DEV,X,Y,ROOTR,ROOTT,ROOTI,ZON,ZOFF,ZTEMP,WORK
      DOUBLE PRECISION ROOTM(NROOTS),ROOTP(NROOTS),Z(1),
     *     DTEMP,SIGZ2M,SIGZ2,ZTNORM,RATIOM,
     *     RATIO,ZMIN,ZMAX,AZTEMP,AZSUM,AZSUMM,SIGN
C
C     MOVE ROOTS TO LOCAL STORAGE.
C
      NROOTS = NON+NOFF
      DO 100 J = 1,NROOTS
      ROOTR(J) = ROOTM(J)
  100 ROOTI(J) = ROOTP(J)
C
C     EXPAND ZEROES ON UNIT CIRCLE
C
      CALL EXPAND(NON,ROOTR(NOFF+1),ROOTI(NOFF+1),NZON,ZON,WORK)
C
C     CALCULATE NUMBER OF COMBINATIONS TO BE TRIED.
C
      NTRY = 2**(NOFF-1)
C
C     TRY THEM.
C
      JFLAG = 1
C
C     SET FILTER INDEX TO 0.
C
      J = 0
C
C     SET INITIAL VALUES TO + INFINITY.
C
      SIGZ2M = 1.D75
      RATIOM = 1.D75
      AZSUMM = 1.D75
C
```

```
C     DECIDE WHETHER TO SET EACH ZERO INSIDE OR OUTSIDE
C          UNIT CIRCLE.
C
  130 DO 160 K = 1,NOFF
      J1 = 1+MOD(J/(2**(K-1)),2)
      GO TO (150,140),J1
  140 ROOTT(K) = 1.D0/ROOTR(K)
      GO TO 160
  150 ROOTT(K) = ROOTR(K)
  160 CONTINUE
C
C     EXPAND ZEROES OFF UNIT CIRCLE.
C
      CALL EXPAND(NOFF,ROOTT,ROOTI,NZOFF,ZOFF,WORK)
C
C     GENERATE COMPLETE FILTER IMPULSE RESPONSE OF LENGTH
C          NZTEMP = NZOFF+NZON-1.
C
      CALL POLYMULT(ZTEMP,NZTEMP,ZON,NZON,ZOFF,NZOFF)
C
C     CALCULATE GAIN OF UNNORMALIZED FILTER.
C
      ZTNORM = 1.D0/DSQRT(DABS(ZTEMP(NZTEMP)))
      GO TO (170,200),JFLAG
C
C     INITIALIZATION.
C
  170 SIGZ2 = 0.D0
      DTEMP = 0.D0
      ZMIN = 1.D75
      ZMAX = 0.D0
C
      DO 180 K = 1,NZTEMP
C
C     NORMALIZE IMPULSE RESPONSE TAP WEIGHTS.
C
      ZTEMP(K) = ZTEMP(K)*ZTNORM
      AZTEMP = DABS(ZTEMP(K))
C
C     FIND AND SAVE MAXIMUM AND MINIMUM WEIGHTS
```

```
C           IN EACH GENERATED IMPULSE RESPONSE.
C
        IF (AZTEMP .LE. ZMAX) GO TO 173
        ZMAX = AZTEMP
        KMAX = K
    173 IF (AZTEMP .GE. ZMIN) GO TO 176
        ZMIN = AZTEMP
        KMIN = K
C
C       ACCUMULATE SUM OF ABSOLUTE VALUES OF TAP WEIGHTS.
C
    176 DTEMP = DTEMP+AZTEMP
C
C       ACCUMULATE SUM OF SQARES OF TAP WEIGHTS.
C
        SIGZ2 = SIGZ2+AZTEMP*AZTEMP
C
    180 CONTINUE
C
C       CALCULATE (NZTEMP-1)*ESTIMATED VARIANCE OF
C            MAGNITUDES OF TAP WEIGHTS.
C
        SIGZ2 = SIGZ2-(DTEMP*DTEMP)/NZTEMP
C
C       CALCULATE RATIO OF LARGEST TO SMALLEST TAP WEIGHTS.
C
        RATIO = DABS(ZTEMP(KMAX)/ZTEMP(KMIN))
C
C       RECORD INDEX NUMBER OF FILTER IF THIS IS SMALLEST
C            RATIO YET ENCOUNTERED.
C
        IF (RATIO .GE. RATIOM) GO TO 183
        RATIOM = RATIO
        MRATIO = J
C
C       RECORD INDEX NUMBER OF FILTER IF THIS IS SMALLEST
C            ABSOLUTE SUM YET ENCOUNTERED.
C
    183 AZSUM = DTEMP/NZTEMP
        IF (AZSUM .GE. AZSUMM) GO TO 187
```

```
          AZSUMM = AZSUM
          JSUMM = J
      187 CONTINUE
C
C         RECORD INDEX NUMBER OF FILTER IF THIS IS SMALLEST
C              VARIANCE YET ENCOUNTERED.
C
          IF (SIGZ2 .GT. SIGZ2M) GO TO 190
          SIGZ2M = SIGZ2
          JMIN = J
C
C         INCREMENT INDEX.
C
      190 J = J+1
          IF (J .LT. NTRY) GO TO 130
C
C         RECOMPUTE OPTIMUM IMPULSE RESPONSE.
C
          GO TO (191,192,193),MODE
      191 J = MRATIO
          GO TO 195
      192 J = JSUMM
          GO TO 195
      193 J = JMIN
      195 JFLAG = 2
          GO TO 130
C
C         CHECK THAT ALTERNATING SUM IS GREATER THAN 0.
C
      200 DTEMP = 0.D0
          SIGN = (-1.D0)**(NZTEMP/2)
          DO 210 K = 1,NZTEMP
          DTEMP = DTEMP+SIGN*ZTEMP(K)
      210 SIGN = -SIGN
          IF (DTEMP .LT. 0.D0) ZTNORM = -ZTNORM
C
C         TRANSFER RESULTS TO OUTPUT ARRAY.
C
          NTERMS = NZTEMP
          DO 220 K = 1,NZTEMP
```

```
  220 Z(K) = ZTEMP(K)*ZTNORM
      RETURN
      END
```

```
      SUBROUTINE EXPAND(NROOTS,ROOTM,ROOTP,NTERMS,Z,WORK)
C
C     HERRING  04 JAN, 1976.
C
C     SUBROUTINE TO EXPAND SET OF ROOTS INTO A POLYNOMIAL
C          WITH REAL COEFFICIENTS.
C
C     ARGUMENTS:
C
C     INPUT:
C
C          NROOTS : NUMBER OF ROOTS GIVEN IN ARRAYS ROOTM, ROOTP.
C          ROOTM  : ARRAY OF MAGNITUDES OF ROOTS.
C          ROOTP  : ARRAY OF PHASE ANGLES (IN RADIANS) OF ROOTS.
C
C          IF ROOTP(J) .NE. 0. OR PI, THE J'TH ROOT IS TREATED AS
C               ONE OF A PAIR OF COMPLEX CONJUGATE ROOTS.
C
C     OUTPUT:
C
C          NTERMS : NUMBER OF COEFFICIENTS IN EXPANDED POLYNOMIAL.
C          Z      : ARRAY OF POLYNOMIAL COEFFICIENTS, IN ORDER OF
C                        ASCENDING POWER.
C          WORK   : TEMPORARY STORAGE ARRAY OF DIMENSION .GE. 2*NROOTS.
C
      DIMENSION ROOTM(1),ROOTP(1),Z(1),WORK(1),Y(3)
      DOUBLE PRECISION ROOTM,ROOTP,Z,WORK,Y,PI
      DATA Y(1)/1.D0/
      PI = DATAN2(0.D0,-1.D0)
      NTERMS = 1
      Z(1) = 1.D0
      DO 500 J = 1,NROOTS
C     MOVE PARTIALLY EXPANDED POLYNOMIAL TO WORK AREA.
      DO 100 K = 1,NTERMS
  100 WORK(K) = Z(K)
      TLW = NTERMS
C     DECIDE WHETHER SINGLE ROOT OR CONJUGATE PAIR.
      IF (ROOTP(J) .EQ. 0.D0) GO TO 200
      IF (ROOTP(J) .EQ. PI) GO TO 300
C     MUST BE CONJUGATE PAIR.
```

```
      Y(2) = -2.DO*ROOTM(J)*DCOS(ROOTF(J))
      Y(3) = ROOTM(J)*ROOTM(J)
      IDY = 3
      GO TO 500
C     ROOT ON POSITIVE REAL AXIS.
  200 Y(2) = -ROOTM(J)
      GO TO 400
C     ROOT ON NEGATIVE REAL AXIS.
  300 Y(2) = ROOTM(J)
  400 IDY = 2
  500 CALL POLYMULT(Z,NTERMS,WORK,IDW,Y,IDY)
      RETURN
      END
```

```
C      SUBROUTINE POLYMULT
C
C      PURPOSE
C         MULTIPLY TWO POLYNOMIALS
C
C      USAGE
C         CALL POLYMULT(Z,IDIMZ,X,IDIMX,Y,IDIMY)
C
C      DESRIPTION OF PARAMETERS
C         Z      - VECTOR OF RESULTANT COEFFICIENTS, ORDERED
C                  FROM SMALLEST TO LARGEST POWER
C         IDIMZ  - DIMENSION OF Z ( CALCULATED )
C         X      - VECTOR OF COEFFICIENTS FOR FIRST POLYNOMIAL ,
C                  ORDERED FROM SMALLEST TO LARGEST POWER
C         IDIMX  - DIMENSION OF X (DEGREE IS IDIMX-1)
C         Y      - VECTOR OF COEFFICIENTS FOR SECOND POLYNOMIAL ,
C                  ORDERED FROM SMALLEST TO LARGEST POWER
C         IDIMY  - DIMENSION OF Y (DEGREE IS IDIMY-1)
C
C      REMARKS
C         Z CANNOT BE IN THE SAME LOCATION AS X
C         Z CANNOT BE IN THE SAME LOCATION AS Y
C
C      SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C         NONE
C
C      METHOD
C         DIMENSION OF Z IS CALCULATED AS IDIMX+IDIMY-1
C         THE COEFFICIENTS OF Z ARE CALCULATED AS SUM OF  PRODUCTS
C         OF COEFFICIENTS OF X AND Y, WHOSE EXPONENTS ADD UP TO
C         THE CORRESPONDING EXPONENT OF Z
C
C
C
       SUBROUTINE POLYMULT(Z,IDIMZ,X,IDIMX,Y,IDIMY)
       DIMENSION X(1),Y(1),Z(1)
       DOUBLE PRECISION X,Y,Z
       IF(IDIMX*IDIMY) 10,10,20
   10  IDIMZ=0
       GOTO 50
```

```
   20 IDIMZ=IDTMX+IDIMY-1
      DO 30 I=1,IDIMZ
   30 Z(I)=0
      DO 40 I=1,IDIMX
      DO 40 J=1,IDIMY
      K=I+J-1
   40 Z(K)=X(I)*Y(J)+Z(K)
   50 RETURN
      END
```

```
      SUBROUTINE NEWPAGE(IUNIT)
C
C     HERRING  05 APRIL, 1977.
C
C     SUBROUTINE TO EJECT PAGE ON DIGITAL DECWRITERII
C          WITH FORMFEED OPTION.
C
      DIMENSION I(21)
      DATA I/8Z0B404040,19*(8Z40404040),8Z40404007/
      IUN = IUNIT
      IF (IUN .LE. 0) IUN = 108
      WRITE (IUN,100) I
      RETURN
  100 FORMAT(21R4)
      END
```